# Tutorial-2

Name : DIVYANSH DUBEY
Section : F

Class roll no - 6
University roll no - 2016739

Q1 What is the time complexity of the code below and how much?

```
void fun (int n)
{
  int j=1, i=0;
  while (i<n) {
    i+=j;
    j++;
  }
}
```

Ans

| | |
|---|---|
| j=1 | i=1 |
| j=2 | i=1+2 |
| j=3 | i=1+2+3 |

] m-level

for (i)

$\because 1+2+3 \text{ ------- } + < n$

$\because \frac{m(m+1)}{2} < n$

$m \approx \sqrt{n}$

By summation method

$\Rightarrow \sum_{i=1}^{m} 1 \Rightarrow 1+1+ \text{ ------- } - \sqrt{n}$ times

$T(n) = \sqrt{n}$

Q2 Write recurrence relation for function that prints Fibonacci series. Solve it to get the time complexity. What will be the space time complexity and why?

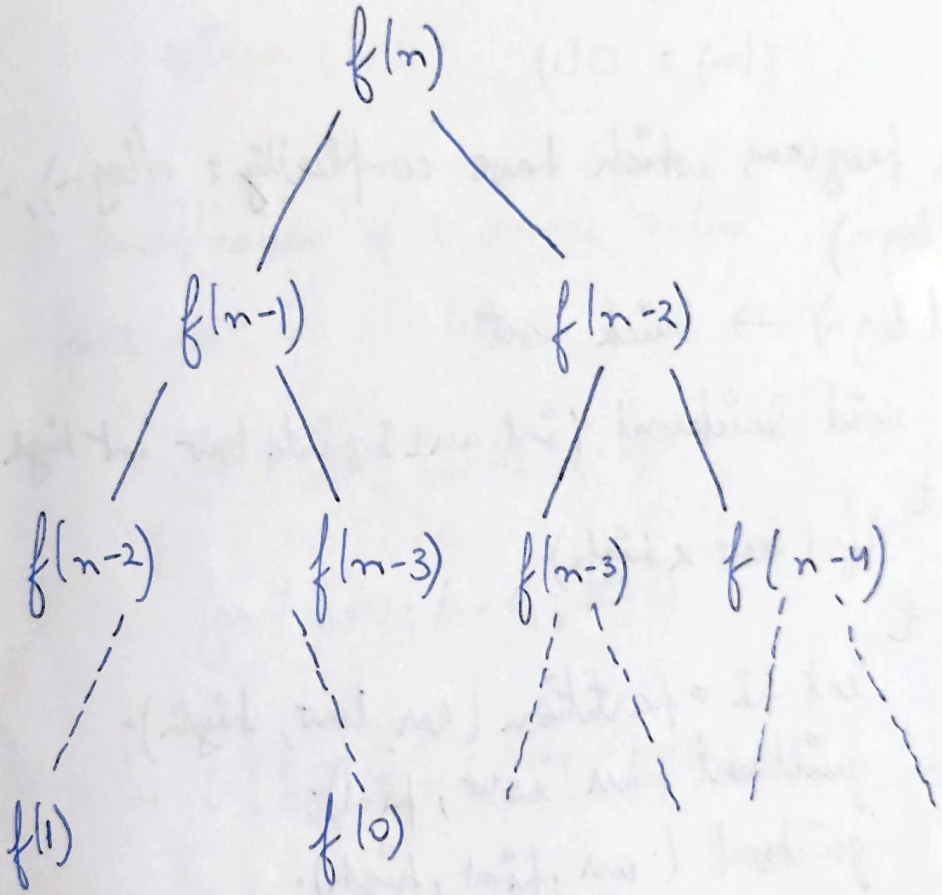
Ans For Fibonacci series

$$f(n) = f(n-1) + f(n-2)$$

$$f(0) = 0$$

$$f(1) = 1$$

By forming a tree




∵ At every function call, we get 2 function calls

for n levels

we have = 2×2 ---- --- n times

$$T(n) = 2^n$$

Maximum space :-

Considering residence

Stack: No. of calls maximum = n

For each call, we have space time complexity O(1)

$$T(n) = O(n)$$

Without considering recursive stack: Each call we have time complexity O(1).

$$T(n) = O(1)$$

Q3 Write programs which have complexity: $n(\log n)$, $n^3$, $\log(\log n)$

Ans $n(\log n)$ → Quick sort

```
void Quicksort (int arr[], int low, int high)
{
  if (low < high)
  {
    int pi = partition (arr, low, high);
    quicksort (arr, low, pi-1);
    quicksort (arr, pi+1, high);
  }
}
int partition (int arr[], int low, int high)
  {
    int pivot, arr[high];
    int i = (low-1);
for (int j=low; j <= high -1; j++)
```

```
    {
      if (arr[i] < pivot)
      {
        i++;
        swap (&arr[i], &arr[j]);
      }
    }
  swap (&arr[i+1], &arr[high]);
    return (i+1);
  }
```

2) $n^3$ → multiplication of 2 square matrix

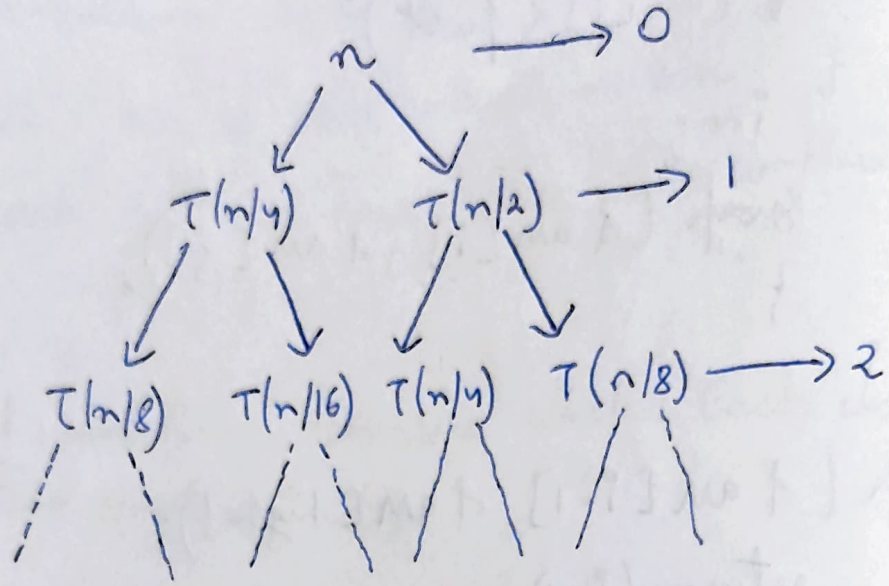
```
for (i=0; i<r1; i++)
  {
    for (j=0; j<c2; j++)
    {
      for (k=0; k<c1; k++)
      {
    res[i][j] += a[i][k] * b[k][j];
      }
```

3) $\log(\log n)$

```
for (i=2; i<n; i = j*i)
  {
    count++;
  }
```

Q4 Solve the following recurrence relation $T(n) = T(n/4) + T(n/2) + T(n/2) + (n^2)$

Ans




At level

$0 \rightarrow Cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{CSn^2}{16}$

$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 n^2 C$

$\vdots$

max level $= \frac{n}{2^k} = 1$

$= k = \log_2 n$

$T(n) = C\left(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \text{------} 5/16\right) \log n^2)$

$T(n) = Cn^2\left(1 + (5/16) + (5/16)^2 + \text{------} (5/16)^{\log n}\right)$

$T(n) = Cn^2 \times 1 \times \left(\frac{1-(5/16)}{1-(5/16)}\right)$

$T(n) = Cn^2 \times \frac{11}{5} \times \left(1 - \left(\frac{5}{16}\right)^{\log n}\right)$

$T(n) = Cn^2 \times \frac{11}{5} \times \left(1 - \left(\frac{5}{16}\right)^{\log n}\right)$

$T(n) = O(n^2 C)$ $\quad O(Cn^2)$

Qs What is the time complexity of the following fun()?

```
int fun(int n){
for(int i=1; i<=n; i++){
for(int j=1; j<=n; j+=1){
// some O(1) task
}}}
```

Ans for

| i | j | j = (n-1)/i times |
|---|---|---|
| 1 | 1 3 | |
| 2 | 1+3+5 | |
| 3 | 1+4+7 | |
| ⋮ | | |
| n | | |

$$\sum_{i=1}^{n} \frac{(n-1)}{i}$$

$$T(n) = \frac{(n-1)}{1} + \frac{(n-1)}{2} + \frac{(n-1)}{3} + \cdots\cdots - \frac{(n-1)}{n}$$

$$T(n) = n\left[1 + 1/2 + 1/3 + \cdots\cdots 1/n\right] - 1 \times \left[1 + 1/2 + 1/3 + \cdots\cdots 1/n\right]$$

$$= n\log n - \log n$$

$$T(n) = O(n\log n)$$

Qs What should be the time complexity of

```
for(int i=2; i<=n; i=pow(i,k))
{
// some O(1)
}
```

where k is a constant

Ans for

$i$
$2^1$
$2^k$
$2^{k^2}$
$2^{k^3}$
$\vdots$
$2^{k^m}$

where
$2^{k^m}$
$k^m = \log_2 n$
$m = \log_k \log_2 n$

$\therefore \sum_{i=1}^{m} 1$

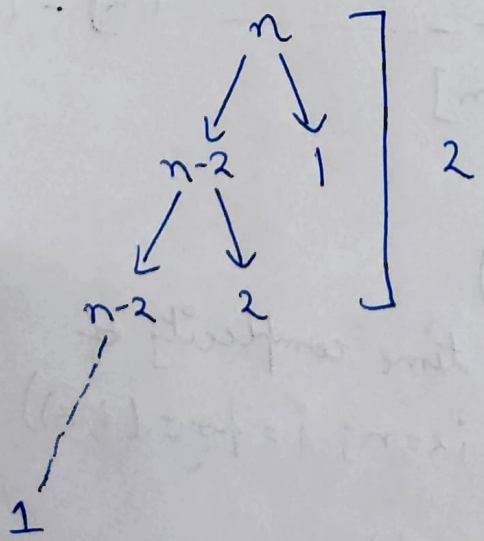
$1+1+1+\cdots\cdots$ m times

$T(n) = O(\log_k \log n)$

Q7 Write a recurrence relation when quick sort repeatedly divides array into 2 parts of 99% and 1%. Derive time complexity in this case. Show the recurrence time while deriving time complexity.

Ans Given algorithm divides array in 99% and 1%

$T(n) = T(n-1) + O(1)$



∴ work is done at each level

$$T(n) = T(n-1) + T(n-2) + \cdots\cdots T(1) + O(1))$$

$$= n \times n$$

$$T(n) = O(n^2)$$

lowest height = 2

highest height = n

Difference = $n-2$ $\quad n>1$

the given algorithm produces linear result.

Q8 Arrange the following in increasing order of rate of growth

a) $n, n!, \log n, \log(\log n), \text{root}(n), n\log n, \log^2, 2^n, 2^{2^n}, 4^n, n$

Ans $1 < \log(\log n) < \log n < \log(n)^2 < \sqrt{n} < n < n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$

b) $2(2^n), 4n, 2n, \log(n), \log(\log(n)), \sqrt{\log(n)}, \log 2n,$

$2\log(n), n, \log(n!), n!, n^2, n\log(n)$

Ans $1 < \log(\log n) < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < 4n < \log(n!) < n^2 < n! < 2^{2n}$

c) $8^{2n}, \log_2(n), n\log(n), n\log_2(n), \log(n!), n!, \log($

$96, 8^{2n}, 7n^3, 5n$

Ans $96 < \log_2 n < \log 2n < 5n < n\log(n) < n\log_2(n) < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$